

**Characterizing Software
Quality Assurance Methods:
*Impact on the Verification of
Learning Systems***

Ali Mili (NJIT) and Frederick
T. Sheldon (ORNL)

Why Deploy an Eclectic Mix of Methods/ Tools/ Techniques?

- *Economic rationale.* Law of diminishing returns.
- *Technical rationale.* Each method is best adapted for some V&V aspect.
- *Pragmatic imperative.* Lack of planning, making use of available resources, tools, skills, etc.
- *Software is a moving target.* Its unclear what method will provide the best return on investment.

Motivation

- Deploying an eclectic mix of methods is more crucial for *online learning systems* than for *traditional systems*.
- No single method is yet known to work (all traditional methods are provably not operational, all new methods admittedly ad-hoc/ partial).
- *Applications of online learning systems are typically critical.*
- *Applications of online learning systems are typically complex.*

Fundamental Question Addressed by Eclectic Approach

- How to formulate verification results in a uniform manner across distinct methods, goals, references?
- How to add verification results established by distinct methods to obtain an aggregate result?
- How to dispatch a complex verification result across several methods, in such a way that each method is delegated the task that it can best handle?

Characterization/Classification Scheme I

- **Goal:** Correctness (completeness, consistency), recoverability preservation, non-functional property.
- **Reference:** Specification against which we are verifying adequacy. possible values: expected function, requirements specification, safety property, test oracle, acceptance oracle, etc.

Characterization/Classification Scheme II

- **Assumption:** All verification methods are based on implicit assumptions, and are meaningful only contingent on these assumptions. Static proofs assume that run-time system is consistent with semantic definition of proof system. Testing assumes that run-time environment is consistent with test environment.

Characterization/Classification Scheme III

- **Certainty.** Some methods claim to establish logical results (that hold with probability 1, contingent upon the assumption). Other methods produce a probabilistic/ stochastic process.
- **Method.** Broad distinction between static and dynamic; possibly others.

Graphical Representation

- Two dimensional table: Goals versus Reference. Entries: assumption, certainty, method.
 - We can revisit several known methods and see if we can characterize them in terms of this classification

Whole Greater than the Sum of its Parts...

- A method can have more than one characterization. Testing a program P on test data T using oracle Ω and finding that P runs successfully on all of T can be interpreted in two ways:
 - correctness of P with respect to $T \setminus \Omega$ (restriction of Ω to T , usually a small specification, in the sense of refinement); or
 - correctness of P with respect to Ω under the assumption that T is a "representative" data set.
- The second interpretation makes a stronger claim, is contingent upon a stronger assumption, hence has a lower associated probability. Which do we choose?
...we don't have to choose if we can add them.

2-D: Goals & Specifications

Goals	Correctness	Other Property	etc
Specifications			
Expected function			
Requirements Specification			
Safety Property			
Test Oracle			
Sub-test Oracle			

How to use the table...

- Lets consider some well known verification methods/properties, and how do we see them fitting into the proposed classification?

Representative Methods/Properties

Certification Testing

- If we submit program P to a test involving test oracle Ω and test data T and we let Ω' be the restriction of Ω to the test data on which the program execution was successful, then the certification test establishes the correctness of the program to the (usually very small) specification Ω'

Representative Methods/Properties

Certification Testing

- This conclusion is conditional on the test environment subsuming (being as demanding as or more demanding) than the operating environment
- We have also established the probable correctness of the program with respect to specification, conditional on the test data being representative of the overall input domain (a doubtful proposition)

Representative Methods/Properties

- ***Static Verification.*** Establishes the correctness of the program with respect to the requirements specification, with probability one (at least in principle), conditional on the operating environment subsuming the semantics inherent in the verification method.
- ***Reliability.*** Establishes the correctness of the program with respect to the requirements specification with some probability P (over some period of time), which can be used to compute such metrics as MTTF by considering factors such as frequency of invocation.

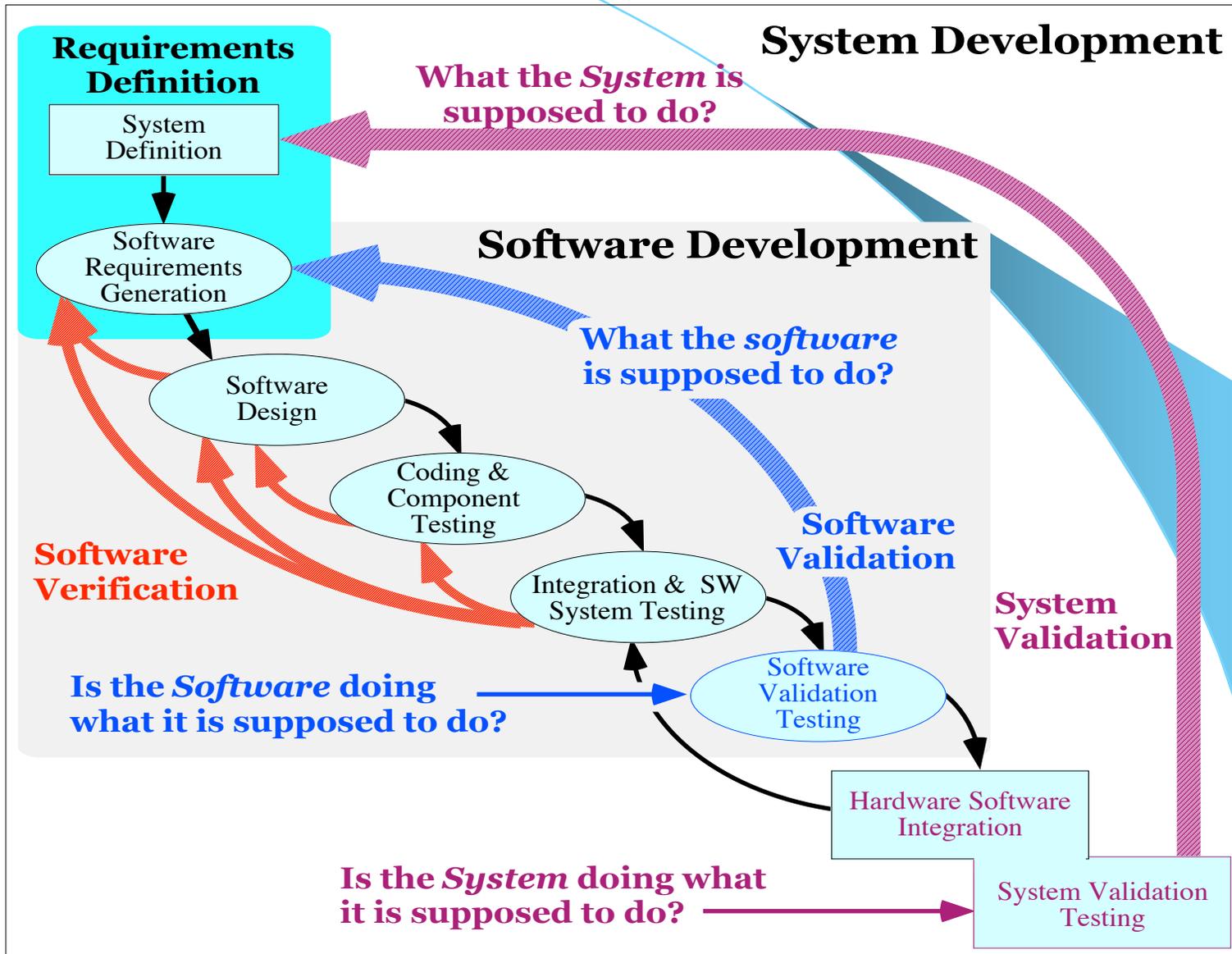
Representative Methods/Properties

- ***Safety***. Safety is nothing more than correctness with respect to a given safety property (e.g., safety critical programs have the requirement, that no single point of failure will cause (the control program) loss of vehicle/stability).
- ***Fault Tolerance***. Fault tolerance is a system's ability to avoid failure after faults have caused errors; it is based on the premise that errors are detected before failure occurs, and is possible only to the extent that recoverability is preserved. Fault tolerance achieves failure avoidance with probability 1, assuming that the fault tolerance infrastructure (code for error detection, error recovery, etc) is trustworthy/ correct.

Representative Methods/Properties

- ***Fail Safety***. Fail safety provides that the program can preserve a safe behavior even when it fails; this requires recoverability preservation with respect to the safety property at hand.
- ***Symbolic Execution***. This technique establishes the correctness of a program with respect to its expected function, with probability one, under the same condition as correctness verification.

Verification



A Calculus of Verification Results

- We can interpret each verification result as establishing that the system under scrutiny S satisfies some goal (that we denote with \geq) wrt some reference R (where R is a requirements specification) contingent upon assumption A (where A is a predicate) with probability p ($0 \leq p \leq 1$), and we write,

$$\text{Prob}(S \geq R \mid A) = p.$$

- For each verification effort, we can summarize our findings by one or more claims of this form.

Usage

- Upon applying several methods, we have cumulated a set of claims of the form:
 $\text{prob}(S \geq_i R_i \mid A_i) = p_i$
- question: what can we claim about system S overall? because this may be intractable in general, we pose instead queries of the form:
 - Can we infer, from this base of facts, a claim of the form:
$$\text{Prob}(S \geq R \mid A) > p$$
for some selected goal \geq , reference R , assumption A , and minimal probability p ?

Sample Identities of this Calculus

$$\text{Prob}(S \geq R_1 + R_2 | A) \geq$$

$$\text{Prob}(S \geq R_1 | A) * \text{Prob}(S \geq R_2 | a),$$

with equality if $S \geq R_1$ and $S \geq R_2$ are statistically independent events, where $R_1 + R_2$ is the join of R_1 and R_2 in the refinement lattice.

- if $A' \implies A$, then $\text{Prob}(S \geq R | A) \geq \text{Prob}(S \geq R | A')$.
- If $R' \geq R$ then $\text{Prob}(S \geq R | A) \geq \text{Prob}(S \geq R' | A)$.

All of these are very simplistic, trivial identities; we will seek tighter identities, that can be useful in an inference.

Prospects / Conclusions

- Derive an *algebra of verification results* using refinement logics, probability theory, etc.
- Consider an *inference system* that supports queries of interest.
- Explore its *application to methods of online learning systems*
 - Cumulate verification results obtained from distinct methods
 - Identify redundancies and complementarily between methods and
 - Infer new verification results from existing results